Implementing Graph-based Search Engine using Sparksee Graph Database

Jie Zhou
Advisor Prof. Jeongkyu Lee
Department of Computer Science and Engineering
University of Bridgeport, Bridgeport, CT

Abstract

The purpose of this project is to implement basic search engine which is based on graph data by using graph database. The data used in this project comes from the Marvel Chronology Project which includes all the characteristics in Marvel Universe and all the issues published by Marvel. For the graph database, we use Sparksee Graph Database, which is one of NoSQL databases for high-performance and scalable graph data management. In Sparksee, all the nodes are linked with edges which are provided in the dataset. The system is implemented by Java language and using Sparksee java API to manipulate all database query..

Introduction

Graph database is now used more often in social network than before. As the high scalability and high retrieve response time many companies which have huge amount of data consider to use graph database to store the dataset that have flexible relationship between each other.

In this project, I create a Marvel Universe database that you can find all the heroes appear in the last few decades and all the issues that have been published so far. Generally, we will use relational database to store, sort, join and retrieve the data but it was not efficient when you have to manage big data. The goal of this project is try to use graph model to store the data and implement the same query functions as relational database can do and proof using graph database is more efficient.

Data Model

Not like other NoSQL database you can input your raw data to your database with column-based or document-based format, before you input any data to your graph database first you should design your own graph model and network which like real social network so you can retrieve data logically and get full benefit of getting data faster. In this program I create two types of nodes, Character node and Issue node which contain all the information of character and issue like the name and ID of them. The graph data is show in figure 1.

When they are created they are given an immutable Sparksee-generated unique identifier, the object identifier (OID). The OID is used to refer the object when using Sparksee APIs to retrieve the data. Sparksee will also given an unique identifier for each attributes you created.

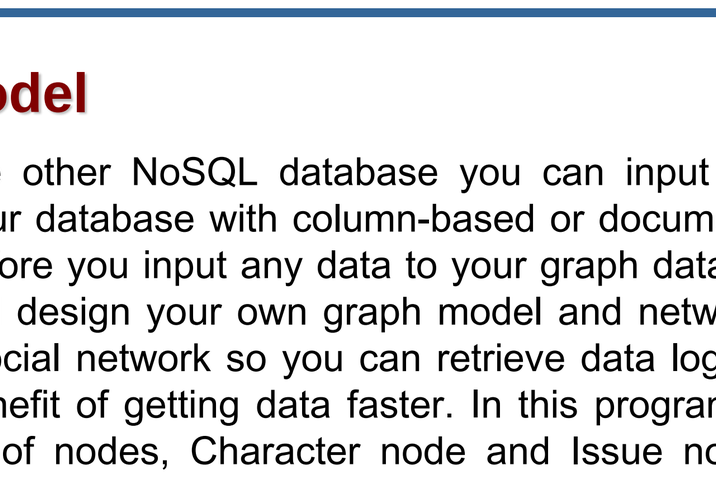

Figure.1 Graph data model

Second, after you input all the node type data you should create or find the relationship between different types of nodes. I use cast type as our edges to link character nodes and issue nodes which means the character appear in the specific issue. All the nodes and edges information you can find on (http://bioinfo.uib.es/~joemiro/marvel.html) webpage. Note that edge not only has unique identifier like node and attribute have but also a neighbor index will be created when you add neighbor node to one specific node.

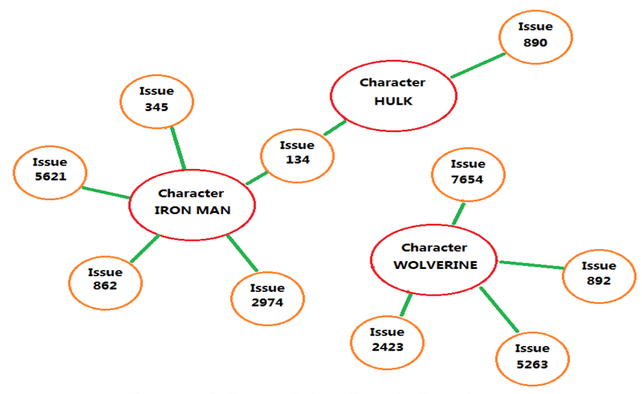

Figure. 2 Part of the Graph Database

Query Test

In this part we use Sparksee java API to test simple query. Sparksee operations accessing the graph through an attribute will automatically use the defined index, significantly improving the performance of the operation. In this example, you can get the node with the specific name attribute you what like get a node name as “WOLVERINE”. After that you can get all the issue node which has edge link to “WOLVERINE”, then you get the all issues that contain character “WOLVERINE”. On the opposite way, you can find the issue node you want by issue id and get the neighbor nodes which type is character node, so you can print all names of the characters who appear in that issue. Also, there is an intersection API you can use to find issue that have both characters who you like most.

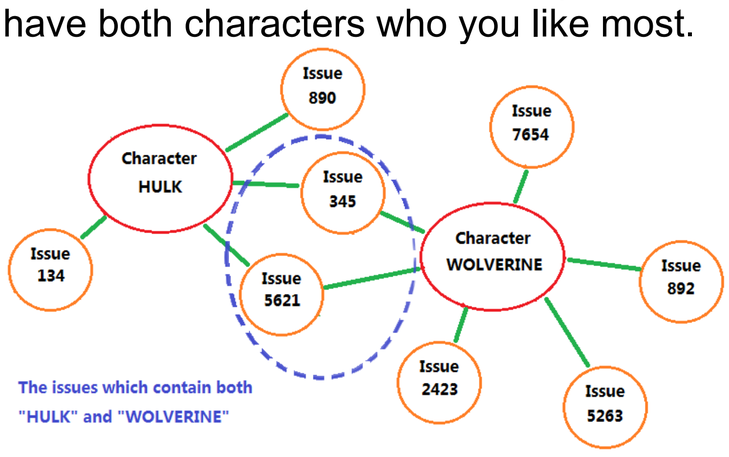

Figure. 3 Example of intersection operation

Conclusion

In this project I upload over 6000 character nodes, over 19000 issue nodes and over 96000 edges. The size of graph database file grows significantly after uploading all the edges. Compare to traditional relational database there are two main differences. First, graph database can support more complex relationship between different type data. This means you can contain more information in the same size of relational database, because relational database use such foreign key to link different tables and it was data redundancy. Second, in graph database you can find specific nodes using index and find the information from neighbor nodes directly, it is faster than using join operations to get the data in relational database.

Future Job

This project only has two types of nodes. If we want to get full benefit of graph database and test the performance we should get more data and create reasonable edges between them.

Multimedia Information Group